ER Simulation

A common application of **queues** is representing lines where people wait for services. These could be physical lines, such as a grocery store line; or they could be virtual lines, such as a printer queue or on-line registration for classes.

In this assignment, you will be simulating the environment of a busy Emergency Room (ER) for a hospital. As patients arrive at the ER, they are placed in a queue to wait to be examined by a doctor. However, in an ER, patients are first screened to determine the severity of their injury or illness to ensure that the most urgent cases are handled first.

For this reason, we will actually be creating three queues: one for patients in *Critical* condition, one for patients in *Serious* condition, and one for patients in *Fair* condition. Critical patients have the highest priority; their injuries are considered life-threatening, and need to be seen as soon as possible. Patients in Serious condition have severe injuries or illnesses, but are relatively stable, and can wait until the Critical patients have been handled. Patients in Fair condition have minor or routine injuries, and will we seen in order after all of the Critical and Serious patients have been seen.

When your program begins executing, it should enter a loop where it reads a line from the data file, interpret it as a command, and then execute it. There are five possible commands:

- **add a patient**: add a patient's name to the appropriate queue. The data line in the file will look like:

```
   add name condition

Example:
   add Trump,D. Critical
```

  You can assume that the name is a string with no spaces in it, and the condition is always either "Critical", "Serious", or "Fair".

- **treat the next patient**: determine who the next patient should be, and remove the name from the queue. The data line in the file will look like:

```
   treat next
```

  Note that you should always treat the patient at the front of the Critical queue first. If the Critical queue is empty, then treat the patient at the front of the Serious queue. If the Serious queue is empty, then treat the person at the front of the Fair queue. If all queues are empty, print the message "No patients are available to treat".

- **treat the next patient in a specified queue**: treat the patient at the front of the specified queue. You can imagine a nurse or intern overriding the queue priorities if he or she lacks a specific skill set to handle critical patients. The data line in the file will look like:

```
   treat condition

Example:
   treat Serious
```

  You can assume that the condition is always either "Critical", "Serious", or "Fair". If the queue is empty, print the message "No patients available in that condition."

- **treat all patients**: treat all of the patients in all of the queues in the order described under "treat next". The data line in the file will look like:

```
   treat all
```

- **exit**: exit the program immediately. The data line in the file will look like:

```
   exit
```

Note:

One form of queue is the **priority queue**, in which each item in the queue is assigned a *priority*. When you dequeue items from the queue, you look first at the items assigned the highest priority and dequeue those first. This means you may dequeue an item that is not at the front of the queue. **We are not using priority queues for this assignment.** Instead, we are creating and managing three separate queues.

Input:

The data file for this program is ERsim.txt. It contains a number of commands in the formats described above.

Output:

Your output should include the following:

- For "add", print a line explaining who you're adding to which queue, and then print out the updated queues. For example, if all three of your queues contained one name each, your output might look something like:

```
>>> Add patient Smith,J. to Serious queue

   Queues are:
   Critical: ['Brown,L.']
   Serious:  ['Smith,J.', 'Martin,B.']
   Fair:     ['Jones,C.']
```

Note that the new name Smith was added to the *left* side of the queue.

- For "treat next", print a line explaining who you're treating and from which queue, and then print out the updated queues. For example, if your queues looked like the queues pictured above, your output might look something like:

```
>>> Treat next patient

   Treating 'Brown,L.' from Critical queue
   Queues are:
   Critical: []
   Serious:  ['Smith,J.', 'Martin,B.']
   Fair:     ['Jones,C.']
```

However, if the queues are all empty, your ouput should look something like:

```
>>> Treat next patient

   No patients in queues
```

- For "treat condition", print a line explaining who you're treating and from which queue, and then print out the updated queues. For example, if your queues looked like the queues pictured above, your output might look something like:

```
>>> Treat next patient on Serious queue

   Treating 'Martin,B.' from Serious queue
   Queues are:
   Critical: []
   Serious:  ['Smith,J.']
   Fair:     ['Jones,C.']
```

However, if the queues are all empty, your ouput should look something like:

```
>>> Treat next patient on Critical queue

   No patients in queue
```

- For "treat all", print output similar to "treat next" for each patient in a queue. For example, if your queues looked like the queues pictured above, your output might look something like:

```
>>> Treat all patients

   Treating 'Smith,J.' from Serious queue
   Queues are:
   Critical: []
   Serious:  []
   Fair:     ['Jones,C.']

   Treating 'Jones,C.' from Fair queue
   Queues are:
   Critical: []
   Serious:  []
   Fair:     []

   No patients in queues
```

- For "exit", simply print a line showing the exit command. For example, your output might look something like:

```
>>> Exit
```